Estimation for Spacecraft Docking with a Known Target

APPROVED BY

SUPERVISING COMMITTEE:

Renato Zanetti, Supervisor

Maruthi Akella

Estimation for Spacecraft Docking with a Known Target

by

James Taggart Bell

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2019

Dedicated to

William H. Burke Jr.

and

Alvie Vernon Bell

Acknowledgments

I would like to thank Dr. Akella and Dr. Zanetti for their continued support, guidance, and patience as my advisors. I must doubly thank each of them, first Dr. Akella for accepting me as graduate student and research assistant, then Dr. Zanetti for being the main advisor for this report. Their trust in me with both research and lab work is fully appreciated, and the knowledge and experiences I gained from this opportunity here at UT will be carried with me throughout the rest of my life.

I must also thank my fellow graduate students, especially those in the NEAR and Auto GNC labs, for their friendship, assistance, and openness in sharing their cultures, ideas, and time with me, and I plan to maintain the friendships and connections I have made for many years to come.

Of course, I would not have made it anywhere near this point without the encouragement and love of my family, who always believed that I could do anything and expected no less of me.

Estimation for Spacecraft Docking with a Known Target

James Taggart Bell, MSE
The University of Texas at Austin, 2019

Supervisor: Renato Zanetti

The problem of autonomous docking in space is difficult, open, and increasingly important. In the case of this report, a chaser spacecraft attempting to dock with the International Docking System Standard on a target craft is analyzed. After motivating the need for augmenting the relative position and orientation estimates with the estimates of their rates, an Extended Kalman Filter is derived to estimate these states with LIDAR and accelerometer measurements. The main assumptions for this report are that the target be in a nearly circular orbit and that the chaser start in near proximity of the target and have access to its own orientation. The filter is shown to work along a predetermined trajectory with both deadbeat control informed by the true state and a closed-loop feedback controller informed by the estimated state.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

When Gemini 6 became the first spacecraft to rendezvous with a target in orbit, it was piloted by a human astronaut. This paradigm for manual docking would reign supreme in the U.S. until March 3rd of this year, when the Crew Dragon spacecraft docked autonomously with the ISS, although the Russians have been doing it to one degree or another since the radar-based system Igla onboard the Soyuz [11]. The reasons for this delay are clear. First, autonomous docking is a hard problem, requiring knowledge of both relative pose and pose rates, as well as the orbital mechanics of each spacecraft. Second, the cost associated with failure is astronomical, as each craft is put at risk of damage. So, rather than risk human capital and lives, space agencies have opted to test these algorithms in labs and between CubeSats. Lab testing is especially inviting, as it allows for the rapid development and deployment of algorithms for docking in a variety of situations, at a many magnitudes lower price. Indeed, NASA's 2015 Robotics, Tele-Robotics and Autonomous Systems Roadmap has Relative Navigation, Autonomous GNC, and Robot Modelling and Simulation all as part of their technological roadmap [17].

Engineers and researchers have been examining the task of orbital ren-

dezvous for a long time, and through that process two main types of docking have emerged: cooperative and non-cooperative. Docking is cooperative when the target spacecraft assists in docking in some way, stereotypically by displaying easily recognized features to assist in navigation while maintaining a constant attitude. A non-cooperative target could be one about which there is little to no prior information, making navigation difficult, or which is either uncontrolled or actively avoiding docking. The International Space Station is equipped with an International Docking System Standard (IDSS) adapter [14], thereby aiding the docking craft in navigation and providing a predetermined interface for actually making the mechanical connection.

The IDSS has features optimized for LIDAR, infrared, and RGB vision systems, which are the main sensors used in docking [18]. Other sensors include RF-based systems and Differential and Carrier Phase Differential approaches [20] [18]. LIDAR systems have the distinct advantage of incorporating range measurements with their angle measurements. Angles-only navigation can be much more difficult, as it requires more measurements to overcome the lack of observability at even moderate distances [6]. Range-only measurements, or 1D LIDAR, can run into observability issues as well [27], meaning that 2D LIDAR is often necessary. 2D LIDAR requires extra data, such as corresponding RGB or infrared 2D data, to determine which points are features from the object model. As a result, a combination of LIDAR and infrared or RGB is often best for docking procedures.

Coming back to the point of determining correspondence between im-

ages of a spacecraft and its model, determining which features between the image correspond to 3D points can range from computationally intensive to intractible, and even in optimal cases is still difficult in orbit [24]. One method for navigation that has taken hold on Earth is combining SLAM plus neural networks to map an object and determine where to dock. This, however, is even more computationally demanding, not to mention the difficulties associated with training a neural net on simulated images [9], [22]. Luckily, the IDSS has infrared markers at geometrically optimal locations for determining pose at multiple scales. This enables high-fidelity measurements for pose acquisition. Once the correspondences are locked down, the process of determining pose from a single measurement is well-researched, if not necessarily easy. Most solutions involve some form of ordinary or total least-squares [4], although the EPnP method [23] and singular value decomposition method presented by Haralick et al. [13] is also viable. Note that these solutions do not give a good understanding of the error covariance, relative velocity, or relative angular velocity, thereby making them somewhat unfavorable for docking.

The desire to incorporate these states for control purposes leads to the utilization of a Kalman Filter [16]. Indeed, a Kalman Filter is not only approximately as accurate for determining pose as the full non-linear optimzation solution, it also rejects noisy measurements and has a relatively low computational complexity. For spacecraft attitude dynamics, a Multiplicative Extended Kalman Filter is often used [7]. Indeed, the main motivation of this paper is to show how to fuse IMU and LIDAR measurements in a MEKF for

the purposes of docking.

Given knowledge of the docking spacecraft pose relative to that of the ISS, plus knowledge of the ISS's orbit, many docking tracjectories can be computed. Of course, these trajectories must meet certain conditions for timeliness, maximum thrust, total fuel consumption, and safety margins. This is essentially a stochastic optimal control problem. If one assumes that the safety margin, or knowledge of relative state, decreases as overall distance decreases, since measurements would be more observable over time, then the stochastic optimal control problem becomes one of optimal control alone. [12] and [25] have shown how to obtain the analytical energy optimal and iterative fuel optimal solutions in the Hill frame for trajectories with known start and end positions and velocities. An extension of the algorithm given in [12] for 2D, 3-point energy optimal trajectories, enables the computation 3D, n-point energy optimal trajectories of arbitrary fidelity. This means that the docking craft can move to a pose from which it can dock head on with the ISS while still maintaining energy optimality, and, given sufficient computation time, fuel optimality [12].

Chapter 2

Background

2.1 Problem Formulation

Beginning at some predetermined location, with some error, we would like to dock with a craft carrying the IDSS adapter using a LIDAR, accelerometer, and known orientation with respect to the ECI frame. The docking trajectory need only maintain attitude such that the target is in view throughout the process. Figure 2.1, below, may be referenced for a general understanding of the geometry of the problem.

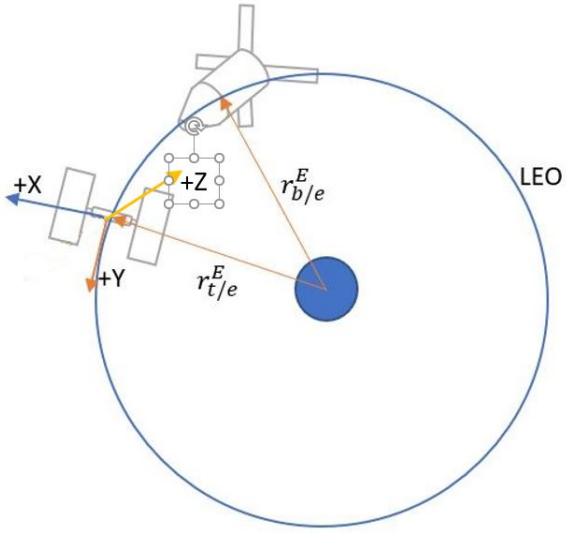

Figure 2.1: Model Orbit

Here, the spacecraft are assumed to be in a counter-clockwise, near-circular orbit. The target is the lead spacecraft, with ECI radius $\mathbf{r}^E_{t/e}$. The axes are fixed to the center of the target frame, with positive X in the radial direction and positive Y in the velocity direction. The chaser spacecraft is in a lag position along the same orbit, with ECI radius $\mathbf{r}^E_{b/e}$, and is pointed toward the target.

2.2 International Docking System Standard

The International Docking System Standard is a document which lays out how docking adapters should be designed for compatibility between various spacecraft, including the International Space Station [19]. The NASA Docking

System (NDS) is NASA's implementation which is to be installed on Orion.

For this report, the pertinent components of the IDSS are the Navigation and Alignment Aids, pictured in Figure 2.2

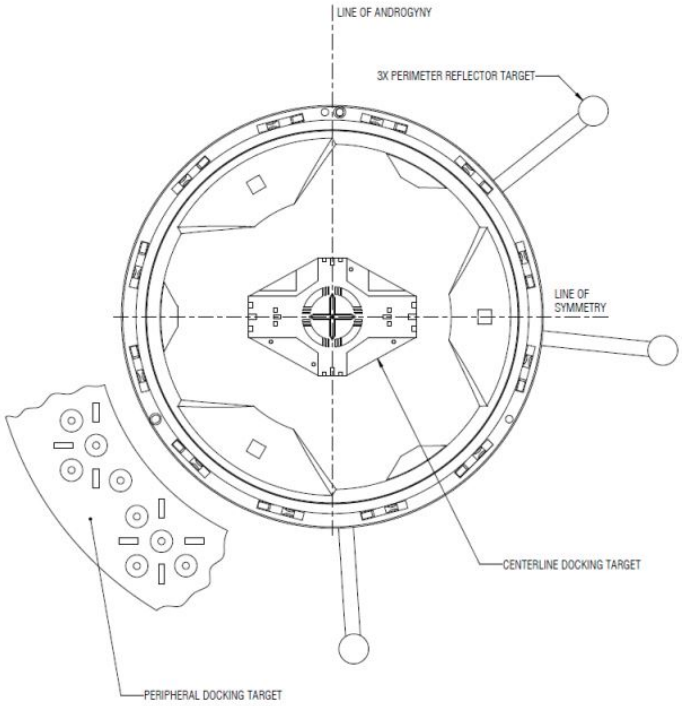

Figure 2.2: IDSS Navigation Aids

The centerline docking target is given in more detail in Figure 2.3. The central cross is in fact offset by .305 meters.

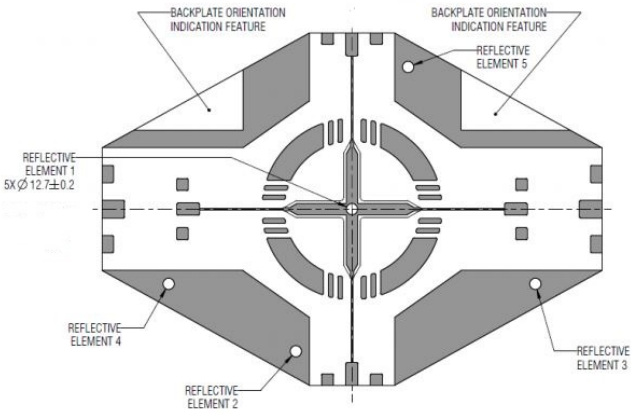

Figure 2.3: IDSS Centerline Docking Target

Table 2.2 provides the locations of the reflectors with respect to the center of the IDSS. The axes are setup assuming it is in the lag position on a spacecraft, with -Y facing outward and +X pointing up.

Table 2.1: IDSS Infrared Reflective Marker Locations

#	X (m)	Y (m)	Z(m)	Component
1	-1.1163	0	.0608	PRT
2	-.1162	0	1.1119	PRT
3	.6888	0	.8806	PRT
4	0	-.305	0	CDT
5	.169	0	.064	CDT
6	-.169	0	-.064	CDT
7	-.08	0	.226	CDT
8	-.08	0	-.226	CDT
9	-.432	0	-.795	PDT
10	-.754	0	-.575	PDT

Docking is performed by interfacing at -1.1 meters in the Y-axis, mean-

ing that for most LIDAR's the time-of-flight will be long enough to still operate properly.

2.3 Clohessy-Wiltshire Equations

To linearize the dynamics, we make use of the Clohessy-Wiltshire approximation [8]. This involves introducing the Hill frame, shown here

$$\hat{\mathbf{i}}_{H/E} = \frac{\mathbf{r}^E_{t/e}}{\left\|\mathbf{r}_{t/e}\right\|} \tag{2.1}$$

$$\hat{\mathbf{k}}_{H/E} = \hat{\mathbf{i}}_{H/E} \times \frac{\mathbf{r}^E_{b/e}}{\left\|\mathbf{r}^E_{b/e}\right\|} \tag{2.2}$$

$$\hat{\mathbf{j}}_{H/E} = \hat{\mathbf{k}}_{H/E} \times \hat{\mathbf{i}}_{H/E} \tag{2.3}$$

$$T^H_E = \begin{bmatrix}\hat{\mathbf{i}}_{H/E} & \hat{\mathbf{j}}_{H/E} & \hat{\mathbf{k}}_{H/E}\end{bmatrix}^T \tag{2.4}$$

The rotation rate of the Hill frame for a prograde orbit is important for computing relative velocities. Here, it is defined in the Hill frame and is given as

$$\omega^H_{H/E} = \begin{bmatrix} 0 \\ 0 \\ -n \end{bmatrix} \tag{2.5}$$

This approximation is known to be accurate only for relative frames with small distances, typically under a kilometer. However, this is fine for our flight regime, as a even the LIDAR onboard Orion, cannot differentiate between fiducial markers at a range of a few hundred meters [5].

2.4 Frames and Vectors

For the purpose of this report, estimated states will be the relative position, relative velocity, target orientation, target angular velocity, and accelerometer bias. This means our state vector will look like

$$\mathbf{x} = \begin{bmatrix} \mathbf{r}_{b/t}^{H} \\ \mathbf{v}_{b/t}^{H} \\ \mathbf{q}_{H\to T} \\ \boldsymbol{\omega}_{T/H}^{T} \\ \mathbf{b}_{a} \end{bmatrix} \tag{2.6}$$

In this report, a superscript H denotes the Hill frame, superscript T denotes the target's body frame, superscript B denotes the Body frame, superscript E denotes the ECI frame, and superscript L denotes the LIDAR frame. The lowercase versions of each letter will denote the center of the coordinate systems for each frame, respectively.

A vector contains a superscript with frame in which it is defined and a subscript with the coordinate system it points to divided by the coordinate system it originates in.

A transformation matrix is given by a capital T, a subscript for the frame in which it is defined, and a superscript for the frame to which it transforms vectors.

An angular velocity vector is defined similarly to a traditional vector, with the added caveat that it relates frames and not coordinates, so the subscript contains capital letters.

In general, quaternions are to defined by a q plus a subscript denoting the rotation it may perform. For example, a quaternion defining the rotation from the Hill to the Target frame is $\mathbf{q}_{H\to T}$. Quaternions are assumed to be vector component first, i.e. $\mathbf{q}_1 = \begin{bmatrix}\mathbf{q}_{1v}\\ q_{1s}\end{bmatrix}$. Quaternion multiplication is given by the well known formula 2.7, and is denoted by the Hamilton Product, $\circledast$ [28].

$$\mathbf{q}_1 \circledast \mathbf{q}_2 = \begin{bmatrix} q_{1s}\mathbf{q}_{2v} + \mathbf{q}_{1v}q_{2s} + \mathbf{q}_{1v} \times \mathbf{q}_{2v} \\ q_{1s}q_{2s} - \mathbf{q}_{1v} \cdot \mathbf{q}_{2v} \end{bmatrix} \tag{2.7}$$

The time evolution of a quaternion due to a constant angular rate is

$$\dot{\mathbf{q}}_{H\to T}(t) = -\frac{1}{2}\mathbf{q}_{H\to T}(t) \circledast \boldsymbol{\omega}^T_{T/H}(t) \tag{2.8}$$

The discrete time solution to 2.8 for a small timestep $\Delta t = t_k - t_{k-1}$ and constant angular velocity $\boldsymbol{\omega}^T_{H/T}(t_{k-1})$ is

$$\begin{aligned} \mathbf{q}_{H\to T}(t_k) &= \mathbf{q}_{H\to T}(t_k) \circledast \begin{bmatrix} sin(\Delta\theta/2)\frac{\boldsymbol{\Delta\theta}}{\|\boldsymbol{\Delta\theta}\|} \\ cos(\Delta\theta/2) \end{bmatrix} \\ \boldsymbol{\Delta\theta} &= \boldsymbol{\omega}^T_{T/H}(t_{k-1})\Delta t \end{aligned} \tag{2.9}$$

For information on how to convert between quaternions and rotation matrices, one may refer to [7] or a variety of other texts and websites.

2.5 Inertial Dynamics

Assuming no non-gravitational forces affecting the orbit of the chaser, such as drag, solar-radiation pressure, or impacts by space debris, the ac-

celerometer truth, $\bar{\mathbf{a}}_b^B(t_k)$, will read only consist of a thrust term. However, a physical acceloremeter also measures bias and white noise, leading to Equation 2.10.

$$\mathbf{a}_b^B(t_k) = \bar{\mathbf{a}}_b^B(t_k) + \mathbf{b}_a(t_k) + \boldsymbol{\nu}(t_k) \tag{2.10}$$

Here, the main terms we are neglecting are non-orthogonality, misalignment, and scale factor matrices, although some models are known to contain hundreds of terms.

The white noise is zero mean and has covariance $\boldsymbol{\sigma}_\nu^2$, which is a function of both sampling time, Δt, and noise density N_0. This is typically called velocity random walk.

$$\boldsymbol{\sigma}_\nu = N_0/sqrt(\Delta t) \tag{2.11}$$

The accelerometer bias is also defined by a random walk model, but typically the noise density is given as a combination of bias instability, $\boldsymbol{\sigma}_{bi}^*$ and the averaging time that minimizes the Allan Variance, t^*. Hence, the standard deviation of the accelerometer bias white noise is

$$\boldsymbol{\sigma}_\upsilon = sqrt(\frac{t^*}{\Delta t})\boldsymbol{\sigma}_{bi}^* \tag{2.12}$$

The accelerometer bias evolves via the following discrete time process.

$$\mathbf{b}_a(t_k) = \mathbf{b}_a(t_{k-1}) + \upsilon(t_k) \tag{2.13}$$

where $\upsilon(t_k)$ is the discrete time random walk for the bias.

2.6 LIDAR

The LIDAR captures both a depth and intensity image. The intensity image is used to find infrared reflectors and assign an elevation, α, and azimuth, β. The depth image provides a range measurement, ρ.

Initially, it is impossible to determine which reflector in the image is correlated to the reflectors in the reference model. For a small number of reflectors in both the image and the model, obtaining these correlations is merely a problem of testing all geometrically feasible combinations and using the best solution, similar to [24]. In the case of the IDSS, the perimeter reflector targets create a scalene triangle, meaning that a best-fit search amongst the features at the greatest distance from the center will yield a pose rather quickly. From there,the rest of the reflectors can be identified and tracked.

Chapter 3

Theoretical Approach

3.1 Dynamics Model

The closed-form solution to the Clohessy-Wiltshire equations do not include a control term. However, given that the control is separate from the state, the following equation can be written via the continuous time CW equations

$$\dot{\mathbf{x}}_{r,v}(t) = A\mathbf{x}_{\mathbf{r,v}}(t) + B\mathbf{u}(t) \tag{3.1}$$

where

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0_{3x3} \\ I_{3x3} \end{bmatrix} \tag{3.2}$$

and n is the mean rate of the spacecraft with respect to the ECI frame.

This style of linear system has a well known solution given by

$$\mathbf{x}_{r,v}(t) = \Phi(t, t_0)\mathbf{x}_{r,v}(t_0) + \int_{t_0}^{t} \phi(t, \tau)B(\tau)\mathbf{u}(\tau)d\tau \tag{3.3}$$

Taking $B(\tau)$ and $u(\tau)$ to be constant between t_0 and t and integrating, equation

3.3 becomes

$$\mathbf{x}_{r,v}(t) = \Phi(t, t_0)\mathbf{x}_{r,v}(t_0) + \Psi(t, t_0)\mathbf{u} \tag{3.4}$$

where, given $t - t_0 = \Delta t$,

$$\Phi(t, t_0) = \begin{bmatrix} 4 - 3cos(n\Delta t) & 0 & 0 & \frac{sin(n\Delta t)}{n} & \frac{2(cos(n\Delta t)-1)}{n} & 0 \\ 6sin(n\Delta t) & 1 & 0 & \frac{2(cos(n\Delta t))}{n} & \frac{4sin(n\Delta t)}{n} - 3\Delta t & 0 \\ 0 & 0 & cos(n\Delta t) & 0 & 0 & \frac{sin(n\Delta t)}{n} \\ 3nsin(n\Delta t) & 0 & 0 & cos(n\Delta t) & 2 * sin(n\Delta t) & 0 \\ 6n(cos(n\Delta t) - 1) & 0 & 0 & -2sin(n\Delta t) & 4cos(n\Delta t) - 3 & 0 \\ 0 & 0 & -nsin(n\Delta t) & 0 & 0 & cos(n\Delta t) \end{bmatrix}$$

$$\Psi(t, t_0) = \begin{bmatrix} \frac{(cos(n\Delta t)-1)}{n^2} & \frac{2(sin(n\Delta t)-n\Delta t)}{n^2} & 0 \\ \frac{2(sin(n\Delta t)-n\Delta t)}{n^2} & \frac{4(cos(n\Delta t)-1+3n^2/2\Delta t^2)}{n^2} & 0 \\ 0 & 0 & \frac{(cos(n\Delta t)-1)}{n^2} \\ \frac{sin(n\Delta t)}{n} & \frac{2(cos(n\Delta t)-1)}{n} & 0 \\ \frac{2(1-cos(n\Delta t))}{n} & \frac{(4sin(n\Delta t)-3n\Delta t)}{n} & 0 \\ 0 & 0 & \frac{sin(n\Delta t)}{n} \end{bmatrix} \tag{3.5}$$

The dynamics governing the orientation of the target spacecraft are assumed to be only partially known. The time evolution of angular velocity of the target with respect to the ECI frame is assumed to be an exponentially correlated random variable (ECRV) with time constant τ_ω. The Hill frame rotates with respect to the ECI frame with rate $\boldsymbol{\omega}^H_{H/E} = \begin{bmatrix} 0 \\ 0 \\ -n \end{bmatrix}$, meaning that the discrete time solution for $\boldsymbol{\omega}^T_{H/T}(t_k)$ is

$$\boldsymbol{\omega}^T_{H/T}(t_k) = e^{-\frac{\Delta t}{\tau_\omega}}(\boldsymbol{\omega}^T_{H/T}(t_{k-1}) - \boldsymbol{\omega}^H_{H/E}) + \boldsymbol{\omega}^H_{H/E} \tag{3.6}$$

This assumes that the target is attempting to maintain a constant orientation in the ECI frame. Other models, such as controlling to a constant Sun or Earth heading could also be assumed.

Finally, given constant time step, the discrete time propagation for the accelerometer bias is given by 2.13

3.2 Measurement Model

The only measurements being used for the filter come from a LIDAR mounted along the $-\mathbf{x}$ body axis. This means that the range, azimuth, and elevation measurements from the LIDAR must be related to the Hill-frame vector from the target to the body, $\mathbf{r}_{b/t}^{H}$ and the quaternion rotation from the Hill frame to the target frame, $\mathbf{q}_{H\to T}$.

First, the relationship between the measurement vector $\mathbf{z} = \begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix}$ and the vector from the LIDAR's focal point to the i-th feature on the target, in the LIDAR frame, $\mathbf{r}_{i/l}^{L}$ can be seen in Figure 3.1

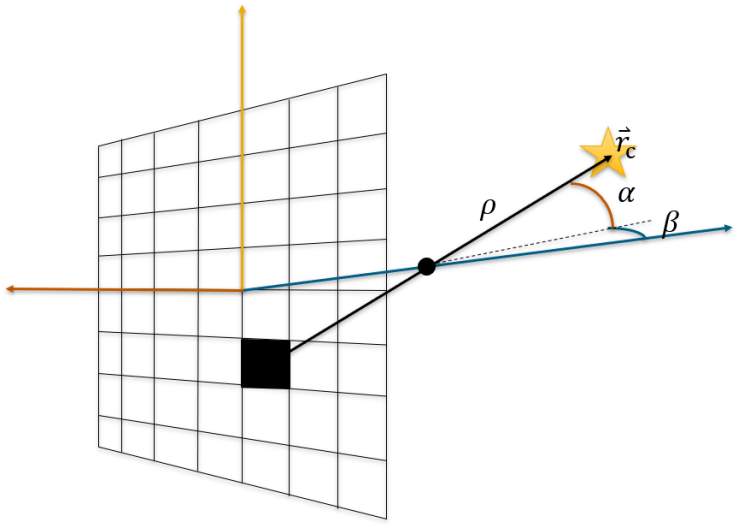

Figure 3.1: Illustration of a LIDAR measurement

From this, one can define

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2 + z^2} \\ \alpha &= tan^{-1}(y/z) \\ \beta &= tan^{-1}(x/z) \end{aligned} \tag{3.7}$$

Recognizing that $\rho = \sqrt{z^2 tan^2(\alpha) + z^2 tan^2(\beta) + z^2(1)}$, we find

$$\mathbf{r}^L_{i/l} = \frac{\rho}{\sqrt{tan(\alpha) + tan(\beta) + 1}} \begin{bmatrix} tan(\beta) \\ tan(\alpha) \\ 1 \end{bmatrix} \tag{3.8}$$

Now, the relationship between $\mathbf{r}^L_{i/l}$ and $\mathbf{r}^H_{b/t}$ can be found via Figure 3.2

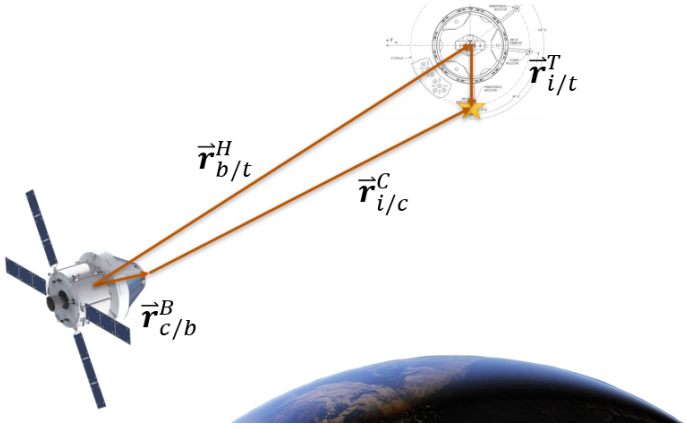

Figure 3.2: Illustration of a LIDAR measurement in the Hill frame

Here, one notes that the both the rotation from the ECI to Body frame and that from ECI to Hill are known, the former from a star tracker and the latter from knowledge of the target's ephemeris. Also, the positions of the LIDAR in the Body frame and the feature in the Target frame are known,

meaning that the feature has already been processed and matched to a 3D model. Initially, through basic geometry, one gets

$$\mathbf{r}_{b/t}^{H} = -T_{HB}(T_{BL}\mathbf{r}_{i/l}^{L} + \mathbf{r}_{l/b}^{B}) + T_{TH}^{T}\mathbf{r}_{i/t}^{T} \tag{3.9}$$

Through a bit of algebra, one can the relate the LIDAR measurement vector back to the state by

$$\mathbf{r}_{i/l}^{L} = T_{BL}^{T}(T_{HB}^{T}(T_{TH}^{T}\mathbf{r}_{i/t}^{T} - \mathbf{r}_{b/t}^{H}) - \mathbf{r}_{l/b}^{B}) \tag{3.10}$$

Hence, with 3.7 and 3.10, the measurement equation $\mathbf{z} = \mathbf{h}(\mathbf{x})$ is fully defined.

3.3 Multiplicative Extended Kalman Filter

At a glance, one can see that, while some of the states have a linear update step, the quaternion propagation is linearized. Furthermore, the measurement update is fully nonlinear. As a result, the basic Kalman Filter is not applicable. However, the Extended Kalman Filter is viable, given that the system is sufficiently linear between measurements. In fact, the EKF is known to be the most optimal linear filter in terms of using the maximum amount of data from each measurement, as it maximizes the inverse of the Fisher Information matrix (gets as close as possible to the Cramer-Rao lower bound).

Quaternions represent an interesting problem for filtering, as they have a constraint state. Merely adding to and normalizing a quaternion does not accurately represent the desired rotation, as resultant rotation, while along the correct direction, will always be less than expected. The multiplicative EKF, or MEKF, is considered to be superior in both computational complexity and theoretical accuracy, as it only operates on the three rotational states and not the fourth constraint state. In this case, the generic MEKF formula is

Propagate

$$\mathbf{x}_k^- = \Phi(t_k, t_{k-1})\mathbf{x}_{k-1}^+ + \Psi(t_k, t_{k-1})\mathbf{u}_{k-1} \tag{3.11}$$

$$\mathbf{q}_k^- = \mathbf{q}_{k-1}^+ \circledast \begin{bmatrix} sin(\Delta\theta/2)\frac{\boldsymbol{\Delta\theta}}{\|\boldsymbol{\Delta\theta}\|} \\ cos(\Delta\theta/2) \end{bmatrix} \tag{3.12}$$

$$P_k^- = \Phi(t_k, t_{k-1})P_{k-1}^+\Phi(t_k, t_{k-1})^T + M(t_k, t_{k-1})QM(t_k, t_{k-1})^T \tag{3.13}$$

Update

$$K_k = P_k^- H_k^T(H_k P_k^- H_k^T + R_k)^{-1} \tag{3.14}$$

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + K_k(\mathbf{y}_k - \mathbf{h}(\mathbf{x}_k)) \tag{3.15}$$

$$\mathbf{q}_k^+ = \mathbf{q}_k^- \circledast \begin{bmatrix} sin(\|\boldsymbol{\delta\theta}\|/2)\boldsymbol{\delta\theta}/\|\boldsymbol{\delta\theta}\| \\ cos(\|\boldsymbol{\delta\theta}\|/2) \end{bmatrix} \tag{3.16}$$

$$P_k^+ = (I - K_k H_k)P_k^-(I - K_k H_k)^T + K_k R_k K_k^T \tag{3.17}$$

where $\boldsymbol{\Delta\theta}$ was defined in 2.9, K_k is the Kalman gain, Q is the error covariance, $M(t_k, t_{k-1})$ are the error partials, and H_k are the measurement

partials. Note that while $\mathbf{q}_k$ is part of the state, it is propagated and updated separately. Also note that the MEKF covariance propagation operates on error states, meaning that one cannot simply take partials of the dynamics/measurements with respect to the states and expect the correct performance.

3.4 Covariance Propagation

Given additive error in the estimate for each state, the propagation of said error is captured by the covariance matrix. In the case of linear systems, such as the position, velocity, angular velocity, and accelerometer bias, the state transition matrix for the error states is the same as that used in state propagation, and can be computed analytically. The same is not quite true for the quaternion update, which has a normalization step. To derive the quaternion error propagation, we begin by defining the error state for quaternions in the MEKF with respect to the estimated states, denoted by hats, and the truth.

$$\mathbf{q}_k = \hat{\mathbf{q}}_k \circledast \delta\mathbf{q} \tag{3.18}$$

where

$$\delta\mathbf{q} = \begin{bmatrix} \delta\boldsymbol{\theta}/2 \\ 1 \end{bmatrix} \tag{3.19}$$

Rearranging 3.19 and taking the time derivative, one gets

$$\delta\dot{\mathbf{q}} = \dot{\hat{\mathbf{q}}}^{-1} \circledast \mathbf{q} + \hat{\mathbf{q}}^{-1}\dot{\mathbf{q}} \tag{3.20}$$

Note, the estimated state evolves through time in the same way as the truth from Equation 2.8

$$\delta\dot{\hat{\mathbf{q}}} = \frac{1}{2}\hat{\mathbf{q}} \circledast \begin{bmatrix} \hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \tag{3.21}$$

Combining 3.20, 2.8, and 3.21:

$$\delta\dot{\hat{\mathbf{q}}} = \frac{1}{2}\begin{bmatrix} -\hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \circledast \hat{\mathbf{q}}^{-1} \circledast \mathbf{q} + \frac{1}{2}\hat{\mathbf{q}}^{-1} \circledast \mathbf{q} \circledast \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} \tag{3.22}$$

Now, using the definition for $\delta\mathbf{q}$ from Equation 3.19 and substituting, we get

$$\delta\dot{\hat{\mathbf{q}}} = \frac{1}{2}\begin{bmatrix} -\hat{\boldsymbol{\omega}} \\ 0 \end{bmatrix} \circledast \delta\mathbf{q} + \frac{1}{2}\delta\mathbf{q} \circledast \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} \tag{3.23}$$

Using 2.7 to further break down 3.23, we find

$$\delta\dot{\hat{\mathbf{q}}} = \frac{1}{2}\begin{bmatrix} -\hat{\boldsymbol{\omega}}\delta q_s - \hat{\boldsymbol{\omega}} \times \delta\mathbf{q}_v + \boldsymbol{\omega}\delta q_s + \delta\mathbf{q}_v \times \boldsymbol{\omega} \\ \hat{\omega} \cdot \delta\mathbf{q}_v - \boldsymbol{\omega} \cdot \delta\mathbf{q}_v \end{bmatrix} \tag{3.24}$$

Defining $\delta\boldsymbol{\omega} = \boldsymbol{\omega} - \hat{\boldsymbol{\omega}}$, 3.24 simplifies nicely to

$$\delta\dot{\hat{\mathbf{q}}} = \frac{1}{2}\begin{bmatrix} \delta\boldsymbol{\omega} + \delta\mathbf{q}_v \times (\boldsymbol{\omega} + \hat{\boldsymbol{\omega}}) \\ -\delta\boldsymbol{\omega} \cdot \delta\mathbf{q}_v \end{bmatrix} \tag{3.25}$$

The last row of 3.25 can be neglected as it is a second order term. Similarly, $\delta \boldsymbol{q}_s = 1$ can be substituted in. Taking the derivative of 3.19 and setting the vector term equal to that of 3.25, it follows that

$$\delta\dot{\boldsymbol{\theta}} = \delta\omega + \delta\mathbf{q}_v \times (2\hat{\omega} + \delta\omega) \tag{3.26}$$

By dropping negligible terms and substituting, it follows that

$$\delta\dot{\boldsymbol{\theta}} = \delta\boldsymbol{\omega} - \delta\boldsymbol{\theta} \times \boldsymbol{\omega} \tag{3.27}$$

Seeing that 3.27 is a 1st-order differential equation in $\delta\boldsymbol{\theta}$, we can solve for the discrete time solution when the error states at time t_k given those at time t_{k-1}:

$$\delta\boldsymbol{\theta}(t_k) = e^{[\hat{\boldsymbol{\omega}}\times]\Delta t}\delta\boldsymbol{\theta}(t_{k-1}) \tag{3.28}$$

Substituting this back into 3.18, we can take derivatives with respect to the error states to get the following partials

$$\frac{\partial\delta\boldsymbol{\theta}(t_k)}{\partial\delta\boldsymbol{\theta}(t_{k-1})} = e^{[\hat{\boldsymbol{\omega}}\times]\Delta t} \tag{3.29}$$

$$\frac{\partial\delta\boldsymbol{\theta}(t_k)}{\partial\delta\boldsymbol{\omega}(t_{k-1})} = I \tag{3.30}$$

Note that each of these is a 3x3 matrix. This implies that the row-count of the covariance matrix is one less than that of the state vector. All other states are merely shifted up once.

Combining this result with Equation 3.5, we can construct $\Phi(t_k, t_{k-1})$, which is needed for Equation 3.13.

$$\Phi(t_k, t_{k-1}) = \begin{bmatrix} \Phi_{r,r}(t_k, t_{k-1}) & \Phi_{r,v}(t_k, t_{k-1}) & 0_{3\times3} & 0_{3\times3} & -\Psi_r(t_k, t_{k-1})T_H^B \\ \Phi_{v,r}(t_k, t_{k-1}) & \Phi_{v,v}(t_k, t_{k-1}) & 0_{3\times3} & 0_{3\times3} & -\Psi_v(t_k, t_{k-1})T_H^B \\ 0_{3\times3} & 0_{3\times3} & e^{[\hat{\boldsymbol{\omega}}\times]\Delta t} & I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3}e^{-\Delta t/\tau_\omega} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{bmatrix} \tag{3.31}$$

The noise propagation matrix $M(t_k, t_{k-1})$, also needed for Equation 3.13, is

$$M(t_k, t_{k-1}) = \begin{bmatrix} \Psi_r(t_k, t_{k-1}) & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ \Psi_v(t_k, t_{k-1}) & 0_{3\times3} & 0_{3\times3} & I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \tag{3.32}$$

The Q matrix contains along its diagonal the accelerometer white noise $\boldsymbol{\sigma}_\nu^2$, bias white noise $\boldsymbol{\sigma}_v^2$, unmodeled torque covariance $\boldsymbol{\sigma}_\omega^2$, and unmodeled force covariance $\boldsymbol{\sigma}_v^2$. These are defined in 4.5.

3.5 Measurement Jacobian

The partial derivative of the measurement equation with respect to the state is called the Jacobian. Since the measurement equation in 3.2 is nonlinear, the first order update is merely an approximation. Once again, the

update step acts on error states, so the partials must be taken with respect to the selfsame error states, which means we will once again need the small angle update from 3.4. The Jacobian is defined as

$$H = \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}}{\partial \mathbf{r}_{i/l}^{L}} \frac{\mathbf{r}_{i/l}^{L}}{\partial \mathbf{x}} = H_r H_x \tag{3.33}$$

Here, H_r is the partial of 3.7 with respect to the LIDAR measurement vector, while H_x is the partial of 3.10 with respect to the error states.

Defining $\mathbf{r}_{i/l}^{L}$ as $[r_x \quad r_y \quad r_z]^T$, one finds the following for H_r

$$H_r = \begin{bmatrix} \frac{\partial \alpha}{\partial r_x} & \frac{\partial \alpha}{\partial r_y} & \frac{\partial \alpha}{\partial r_z} \\ \frac{\partial \beta}{\partial r_x} & \frac{\partial \beta}{\partial r_y} & \frac{\partial \beta}{\partial r_z} \\ \frac{\partial \rho}{\partial r_x} & \frac{\partial \rho}{\partial r_y} & \frac{\partial \rho}{\partial r_z} \end{bmatrix} \tag{3.34}$$

The partials are straightforward:

$$\frac{\partial \alpha}{\partial r_x} = 0 \tag{3.35}$$

$$\frac{\partial \alpha}{\partial r_y} = \frac{r_z}{r_z^2 + r_y^2} \tag{3.36}$$

$$\frac{\partial \alpha}{\partial r_z} = -\frac{r_y}{r_z^2 + r_y^2} \tag{3.37}$$

$$\frac{\partial \beta}{\partial r_x} = \frac{r_z}{r_z^2 + r_x^2} \tag{3.38}$$

$$\frac{\partial \beta}{\partial r_y} = 0 \tag{3.39}$$

$$\frac{\partial \beta}{\partial r_z} = -\frac{r_x}{r_z^2 + r_x^2} \tag{3.40}$$

$$\frac{\partial \rho}{\partial \mathbf{r}_{i/l}^{L}} = \frac{\mathbf{r}_{i/l}^{L}}{\left\| \mathbf{r}_{i/l}^{L} \right\|} \tag{3.41}$$

Getting H_z is slightly more complicated, as $\frac{\partial T_{TL}^T}{\partial \delta \boldsymbol{\theta}}$ is not immediately intuitive. However, the solution when the rotation matrix is multiplied by a vector is well known, yielding the solution:

$$H_z = \begin{bmatrix} \frac{\partial \mathbf{r}_{i/l}^L}{\delta \mathbf{r}_{b/t}^H} & \frac{\partial \mathbf{r}_{i/l}^L}{\delta \boldsymbol{\theta}} \end{bmatrix} = \begin{bmatrix} T_{LB}T_{BL} \\ T_{LB}T_{BL}[(T_{TL}r_{i/t}^T \times] \end{bmatrix} \tag{3.42}$$

Chapter 4

Simulation Setup

4.1 Simulation Goal

The expressly written simulation goal is a subset of the problem outlined in 2.1 - follow docking trajectory from lag 20 meters out in LEO. There are two trajectory cases. For the first trajectory, the chaser position, velocity, and rotation are all assigned, while the target state is simulated via two-body and rotational dynamics. For the second trajectory, the chaser position and velocity are simulated via two-body as well, with control being prescribed by a PD controller attempting to follow trajectory 1. This is meant to illustrate that the estimate is sufficiently accurate for a feedback loop.

4.2 Dynamics

Truth is simulated via simplified force and torque models. The force model is

$$\mathbf{F} = m\big(-\mu\frac{\mathbf{r}^E}{r^E} + \mathbf{u}_F\big) \tag{4.1}$$

where m is mass. Of course, for the target the control is zero. Also, this only applies to the body in the case of Trajectory 2.

The torque model governs how the target is oriented and rotates at any

given time. The torque model is

$$\tau = J^{-1}\big(\mathbf{u}_\tau + \boldsymbol{\omega} \times (J\omega)\big) \tag{4.2}$$

Then, the orientation quaternion is updated via Equation 2.8. For the purpose of docking, the target keeps its body axis oriented with the Hill frame. The Hill frame rotates with constant angular velocity, $\omega^H_{H/E}$, defined in 2.5. For a circular orbit, this is perfect, but for an orbit with low eccentricity, it is merely an approximation, creating some small errors. This is how errors are added into the system for angular rate and, by extension, orientation.

The chaser merely maintains pointing direction towards the center of Hill frame. The DCM for the optimal pointing direction is generated by calculating the following unit vectors

$$\hat{\mathbf{k}}^*_{C/E} = \frac{\big(\mathbf{r}^E_{b/e} - \mathbf{r}^E_{t/e}\big)}{\big\|\mathbf{r}^E_{b/e} - \mathbf{r}^E_{t/e}\big\|} \tag{4.3}$$

$$\hat{\mathbf{i}}^*_{C/E} = \frac{\mathbf{r}^E_{b/e}}{\big\|\mathbf{r}^E_{b/e}\big\|} \times \hat{\mathbf{k}}^*_{C/E} \tag{4.4}$$

$$\hat{\mathbf{j}}^*_{C/E} = \hat{\mathbf{k}}^*_{C/E} \times \hat{\mathbf{i}}^*_{C/E} \tag{4.5}$$

These are then transposed and stacked into the DCM as such

$$T^C_E = \begin{bmatrix} \hat{\mathbf{i}}^*_{C/E} & \hat{\mathbf{j}}^*_{C/E} & \hat{\mathbf{k}}^*_{C/E} \end{bmatrix}^T \tag{4.6}$$

Finally, the optimal quaternion describing the rotation from the ECI frame to the Body frame is

$$\mathbf{q}_{E\to B} = R(T_L^B T_E^L) \tag{4.7}$$

where the $R(\cdot)$ function converts a quaternion to an equivalent DCM [7].

The state as prescribed by equation 2.6 is calculated from the states of the body and target spacecraft. First, we get the rotation from the ECI frame to the Hill frame with 2.4.

Then, $\mathbf{r}_{b/t}^H$ is simply

$$\mathbf{r}_{b/t}^H = T_E^H\left(\mathbf{r}_{t/e}^E - \mathbf{r}_{b/e}^E\right) \tag{4.8}$$

Recalling that the Hill frame is rotating with a constant rate, the velocity, $\mathbf{v}_{b/t}^H$, is

$$\mathbf{v}_{b/t}^H = T_E^H\left(\mathbf{v}_{t/e}^E - \mathbf{v}_{b/e}^E\right) + \omega_{H/E}^H \times \mathbf{r}_{b/t}^H \tag{4.9}$$

The quaternion relating the Hill frame to the Target frame is

$$\mathbf{q}_{H\to T} = q(T_E^T T_E^L) \tag{4.10}$$

where the equation to take a DCM to a quaternion is also given in Crassidis [7].

The angular velocity of the target, defined in the Target frame, is simply

$$\boldsymbol{\omega}_{T/H}^T = T_E^H \boldsymbol{\omega}_{T/E}^T - \omega_{H/E}^H \tag{4.11}$$

Finally, the accelerometer bias is given by 2.13.

4.3 Accelerometer Simulation

For Trajectory 1, the zero-order hold control input over the interval $[t_{k-1}, t_k)$, denoted by $\mathbf{u}(t_k)$, is not precomputed and must be obtained indirectly as $\mathbf{u}_c w(t_k)$ via the following equation

$$\mathbf{u}_c w(t_k) = (\Psi^T(t_k, t_{k-1})\Psi(t_k, t_{k-1}))^{-}1\Psi^T(t_k, t_{k-1})\left(\begin{bmatrix}\mathbf{r}^H_{b/t}(t_k)\\ \mathbf{v}^H_{b/t}(t_k)\end{bmatrix} - \Phi(t_k, t_{k-1})\begin{bmatrix}\mathbf{r}^H_{b/t}(t_{k-1})\\ \mathbf{v}^H_{b/t}(t_{k-1})\end{bmatrix}\right) \tag{4.12}$$

where $\Phi(t, t_{k-1})$ and $\Psi(t, t_{k-1})$ come from Equation 3.5. Note that the system is overdetermined, which is why the least squares solution is used. This approximation is valid for a discrete time trajectory, so long as the orbit is nearly circular. This can be seen in Figure 4.1. Note that the nominal control input is on the order $1e-4$. The differences are most likely due to nonlinearities which are not accounted for by the CW equations.

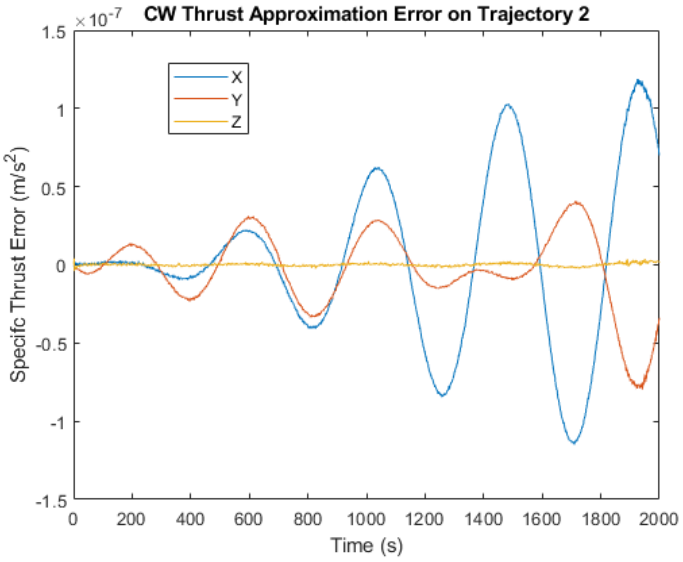

Figure 4.1: Unnormalized Error in Specific Thrust

This is then rotated into the Body frame with T_H^B to get the true acceleration, $\bar{\mathbf{a}}_b^B(t_{k-1}, t_k)$.

For Trajectory 2, the control input is computed via a PD controller, and as such needs only be rotated into the body frame by T_E^B, which is known since we know what T_E^H and T_H^B both are from equations 4.7 and 2.4.

Then, the discrete time white noise is computed with Equation 2.11 and the Cholesky decomposition discussed in Section 4.5. The accelerometer bias random walk is computed from 2.13. Lastly, these are added together to get the simulated accelerometer measurement, given by Equation 2.10.

4.4 Controller

Trajectory 1, is simply given a linear velocity trajectory discretized in time.

For Trajectory 2, the controller is slightly more complicated. Initially, I had tried to follow an optimal trajectory by calculating an optimal control history. However, given an inaccurate initial state this was relatively difficult, so I simply decided to try and track a linear trajectory, Trajectory 1. The errors are first given in the Hill frame then cross correlated by using the control matrix from Equation 3.5. Without the cross-correlation, this fails miserably. Then, a simple PD controller is applied, where the gains were found via the Ziegler-Nichols method [29]. While clearly suboptimal, I figure that if a hacked together PD controller can follow a straight line in the Hill frame using the

estimated state as the input, then the estimated state is sufficiently accurate.

Note that an attitude controller is not used in this case for either the target or chaser. The target is instead merely given an initial rotation rate. If it had a true controller, then the chaser would need a true trajectory controller to attempt to perform docking, which is outside the scope of this report. The chaser does not need an attitude controller because its attitude is not being estimated, so to create one would be unfruitful at best.

4.5 Simulation Parameters

Of course, there are many other parameters that go into a simulation which can affect its performance. The values for some are merely reasonable approximations of physical systems, while others have a well-reasoned behind them. A short discussion of these parameters follows.

All simulations were run in MATLAB. The simulations for Trajectory 1 and 2 were both run with the same random number generator seed at 100 Hz. The Monte Carlo simulation began with the same seed, but of course all subsequent runs had different random number generation, as the seed was not reset.

The initial and process noise covariance matrices are diagonally populated with values as shown in Table 4.5, below

The process noise for position and orientation is zero because it is reasonable to assume that all process noise is induced by unmodeled forces and

Table 4.1: Initial and Process Noise Covariances for Simulation

State	Initial Covariance	Process Noise
$\mathbf{r}_{b/t}^{H}$	1×10^{-5}	0
$\mathbf{v}_{b/t}^{H}$	1×10^{-2}	2.2×10^{-12}
$\mathbf{q}_{H \to T}$	1×10^{-6}	0
$\boldsymbol{\omega}_{T/H}^{T}$	1×10^{-10}	1×10^{-12}
$\mathbf{b}_a$	2.5×10^{-3}	4×10^{-7}

torques, which would act on the velocity and angular velocity states. The values for the accelerometer-related covariances are taken from the datasheet for the sensor used in the lab. Unlisted here is the white noise for the accelerometer measurement, whose covariance is 2.5×10^{-4}. To get the initial error, one performs a Cholesky factorization of the covariance matrix, transposes the result, and multiplies it by a random vector drawn from a normal distribution. This gives a random vector drawn from a normal distribution with the desired covariance. Conveniently, MATLAB provides functions to perform both of these tasks.

Do note that the covariance for the orientation of the target is given in the form of euler angles, not a quaternion. To get the initial error, one must not only obtain the random vector of correct distribution, but then convert it to an error quaternion, as in Equation 3.18.

Camera field of view is also rather important. The majority of runs, including the Monte Carlo runs, were performed with a field of view of 60°. Initially, this was just to show that the estimation still worked as features

dropped out upon approach, but it served the added benefit of neglecting highly nonlinear measurements, as will be discussed later.

Chapter 5

Results

5.1 Trajectory 1

A plot of Trajectory 1 and the estimate of it are given in 5.1. The out-of-plane, or Z, component is not pictured here, but the truth is zero for the whole trajectory. The plot of the required control to generate this trajectory, 5.2, as calculated by 4.12, belies the small radial thrust due to scale, but it is indeed there.

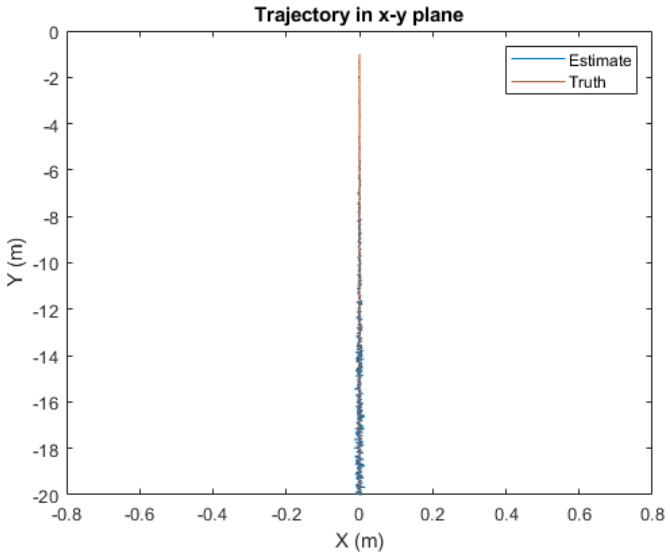

Figure 5.1: Trajectory 1

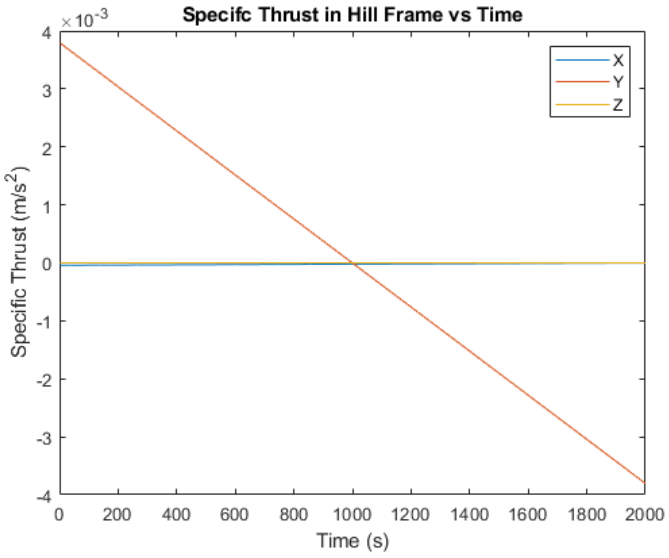

Figure 5.2: Specific Thrust Profile for Trajectory 1

Note that the final position is 1 meter away from the docking target, as mentioned in 2.2.

The error in the position can be found in 5.3.

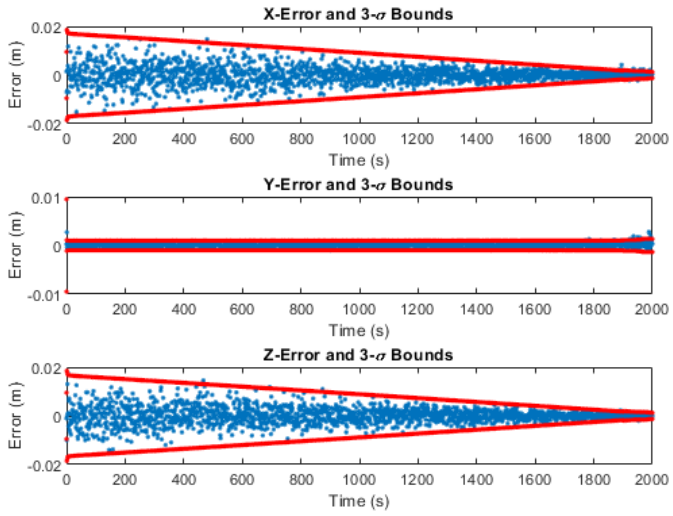

Figure 5.3: Trajectory 1 Position Error

As can be intuited from 5.1, the estimates for radial and out-of-plane position improve with nearness to the docking adapter. This is because the horizontal dilution of precision is higher for a spread out constellation of markers than a clustered one. The errors in a clustered set of measurements tend to require more measurements to average out than those in a spread out set. This can be colloquially referred to as observability.

Less intuitively, the estimate for the along-orbit, or transverse, direction has consistent error and confidence. This could be an artifact of the calculation for noise in the range direction being constant, but also may be related to the linearization not taking into account nonlinearities in the range equation properly.

At the end of the trajectory, all three states have some outliers outside the 3-σ bounds. This is due to the EKF being overconfident in its estimate in the presence of nonlinearities, and is discussed more in 5.4

The velocity estimation error, seen in Figure 5.4, lies more fully within the bounds.

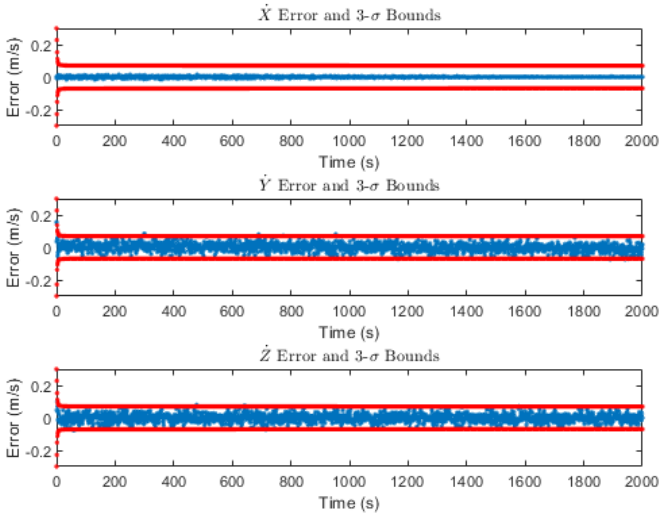

Figure 5.4: Trajectory 1 Velocity Error

The lack of outliers is most likely due to the low nonlinearities in dynamics and the fact that velocity is not directly dependent on the nonlinear measurements, meaning it is smoothed a bit. It is unclear why the erros in the radial direction seem to lie within a much smaller bounds than the covariance suggests they should be in. One possible reason is that the nonlinearities in the lag position of a nearly-circular orbit actually work in the favor of the

Clohessy-Wiltshire approximation for the radial velocitiy, however this is just a conjecture.

Orientation error is given by the euler angles representing the rotation between the estimate and truth, and can be viewed in 5.5. The discontinuities in covariance are caused by measurements leaving the field of view as the chaser approaches the target.

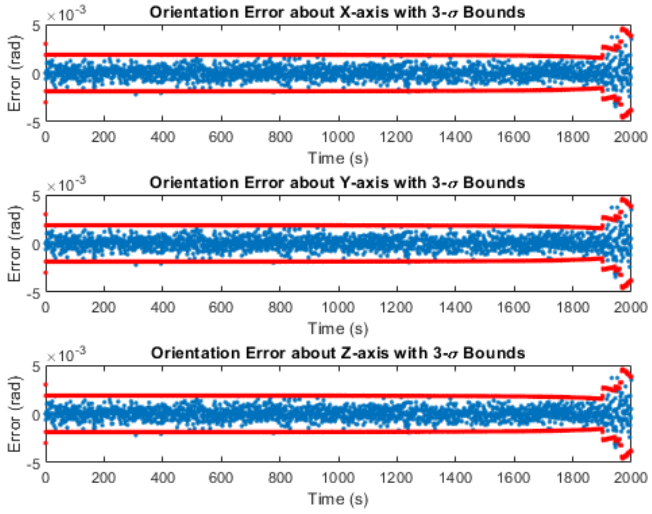

Figure 5.5: Trajectory 1 Orientation Error

The angular velocity error, Figure 5.6, remains within its 3-σ bounds and is zero mean and white.

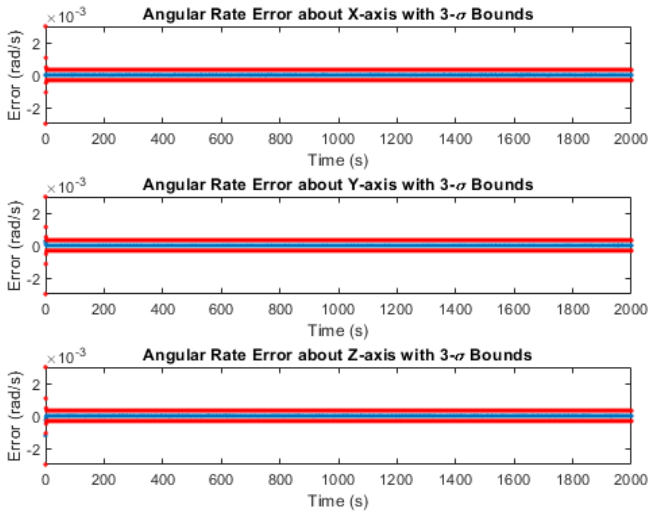

Figure 5.6: Trajectory 1 Angular Velocity Error

The errors do not fill the bounds because the only errors in angular velocity caused by the dynamics are due to the slightly elliptical orbit of the International Space Station and the resulting difference between mean rate and true rotation rate of the Hill frame. However, the process noise is artificially expanded to allow for controlling the space station attitude. When increased by too much, the angular velocity process noise indirectly inflates the orientation error covariance, which negatively affects the orientation estimate by creating structure in the error residuals.

The bias estimation error is given in Figure 5.7. This appears to to track the acceleromter bias, seen in Figure 5.8, rather well, although it is not quite white. This could be due to the tendency for bias to absorb some of the unmodeled dynamics and estimation nonlinearities. However, the fact that it

remains within the 3-σ bounds is a positive.

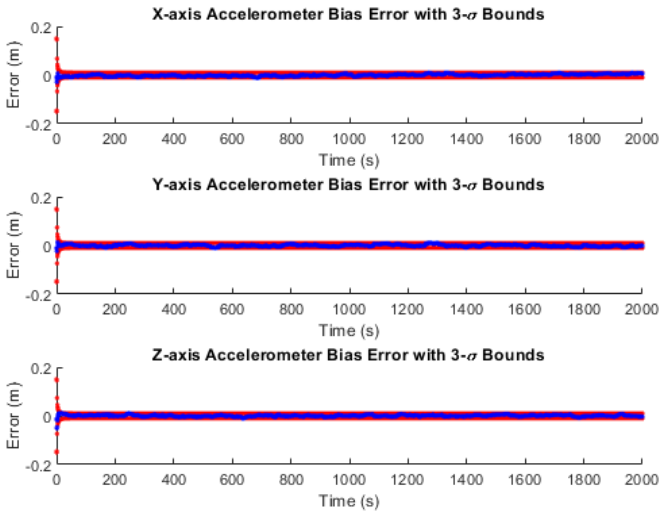

Figure 5.7: Trajectory 1 Accelerometer Bias Error

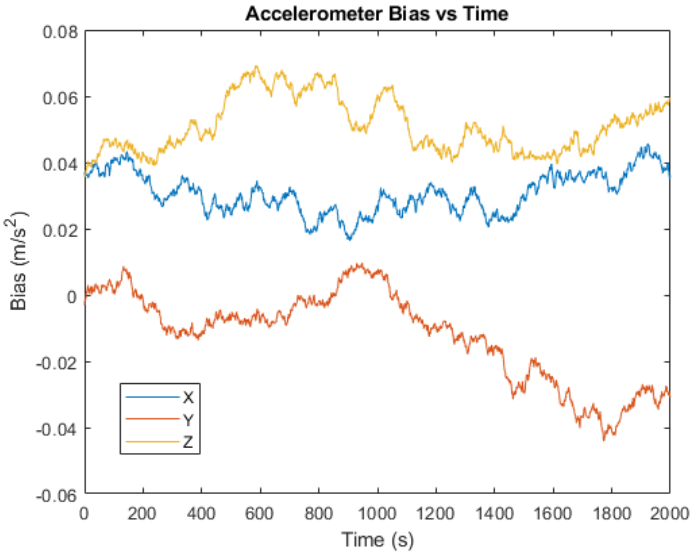

Figure 5.8: Trajectory 1 & 2 Accelerometer Bias

The innovation residuals and covariances are plotted in Figure 5.9. The overlapping covariances are due to residuals from different markers all being plotted on the same graph. These have been left in to better illustrate how the covariances change with range to different markers. It will be concluded in 5.4 that the outermost bound in both the azimuth and elevation plots is actually marker #4 in 2.2.

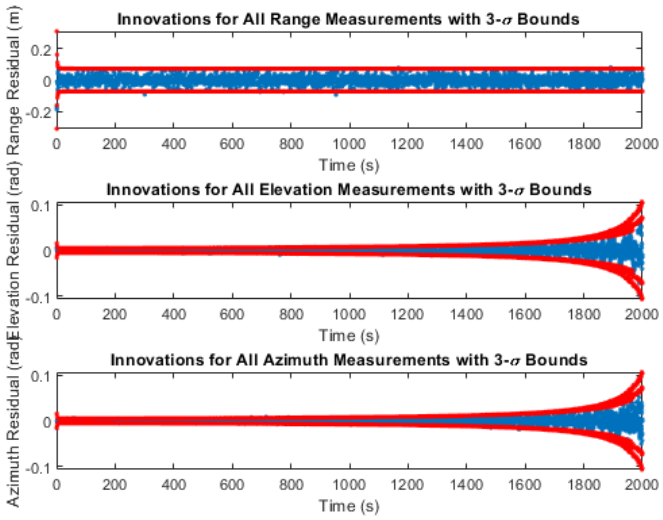

Figure 5.9: Trajectory 1 Innovation Residuals

5.2 Monte Carlo

The fact that the residuals for Trajectory 1 are zero mean and white is promising. To further validate these results, a Monte Carlo simulation was performed with 150 runs. All plots contain the sample mean and unbiased sample covariance. The hope is that the sample covariance will match the

estimated covariance in 5.1 and that the sample mean is close to zero.

The position error plot is given in Figure 5.10.

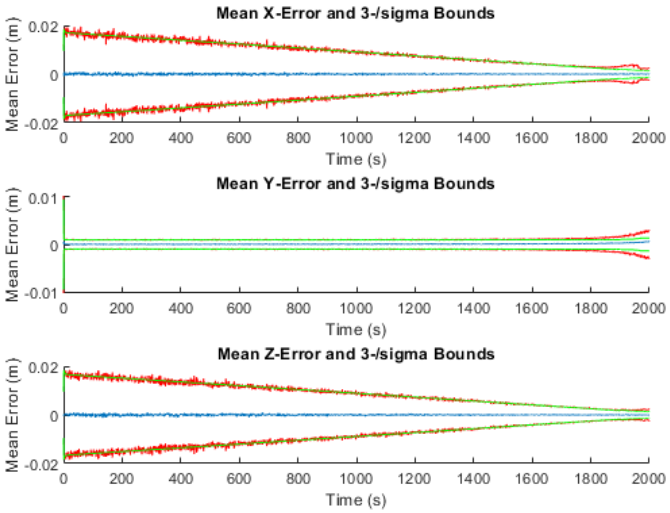

Figure 5.10: Monte Carlo Position Error

Indeed, it can be seen that the sample covariance, red, matches the estimated covariance, green, almost exactly. Also, the mean-square error of the sample mean is $6 \times 10^{-}6$ m^2 for X and Z, and $3 \times 10^{-}6$ m^2, a ratio which matches what would be expected given their relative covariances over the trajectory.

The velocity error plot, 5.11, is equally as promising.

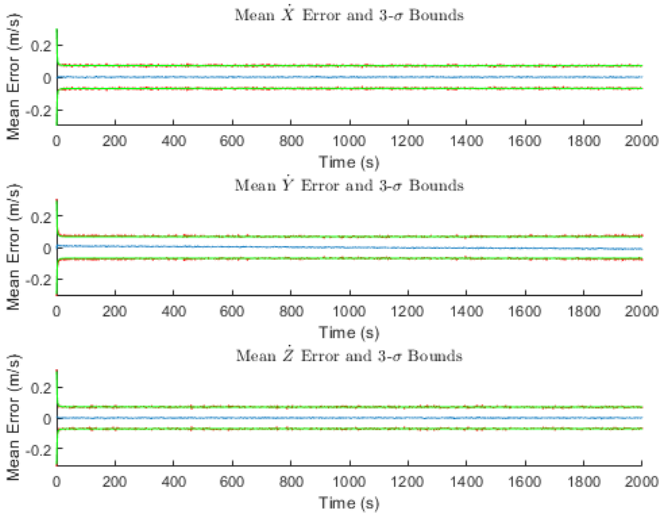

Figure 5.11: Monte Carlo Velocity Error

Similar to 5.10, the sample and estimated covariances match, and the mean-square error is approximately zero with the mean being distributed about zero.

The orientation error plot, 5.12, has some interesting effects toward the end when measurements start to drop out and nonlinearities begin to build up. The outliers seen in 5.5 can be more clearly seen by the sample covariance lying outside the estimated covariance. However, the error is still zero mean.

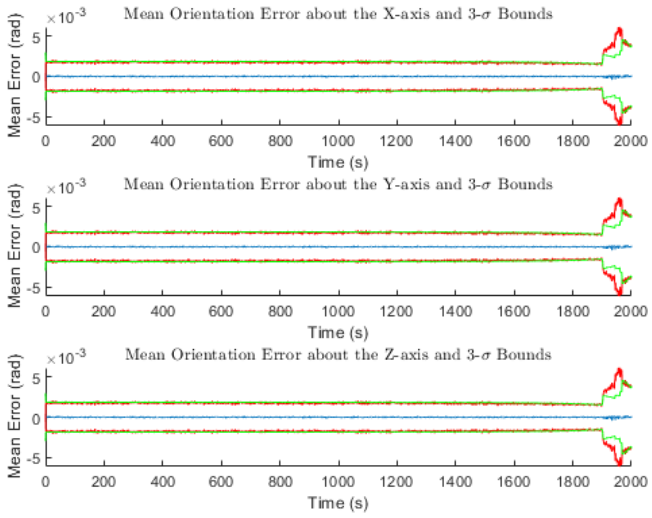

Figure 5.12: Monte Carlo Orientation Error

Time did not permit an analysis of the residuals or bias, but based on the orientation error plot, it can be reasonably assumed that the same trend of repeatibility shown by the Monte Carlo simulation up to this point would persist.

5.3 Trajectory 2

Ultimately, the goal for estimation in docking is to provide feedback for a controller. To this end, Trajectory 2 does just that. It attempts to follow Trajectory 1 with a PD controller.

The resultant trajectory and specific thrust are given in Figures 5.13 and 5.14. In this case, the plotted thrust is the zero-order hold command sent

to the dynamics simulation.

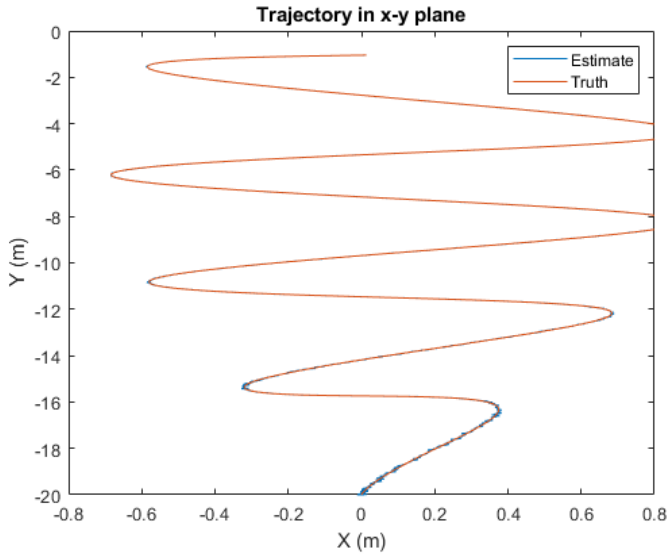

Figure 5.13: Trajectory 2

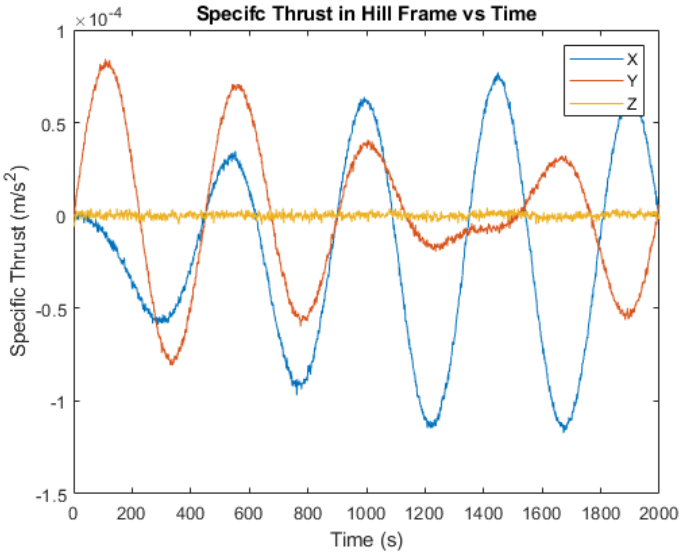

Figure 5.14: Specific Thrust Profile for Trajectory 2

Clearly, the final approach is sub-optimal. This is mainly due to a poorly tuned, feedback-only controller operating linearly in a nonlinear space, as feeding back the estimate or truth states gives the same trajectory. However, because the pointing direction is maintaing the whole time, the docking should still be successful. The hope is that the reader will forgive the small amount of time spent optimizing the controller due to the quality of the estimation along such a poor trajectory.

The position and velocity estimation errors are given in the following plots, 5.15 and 5.16.

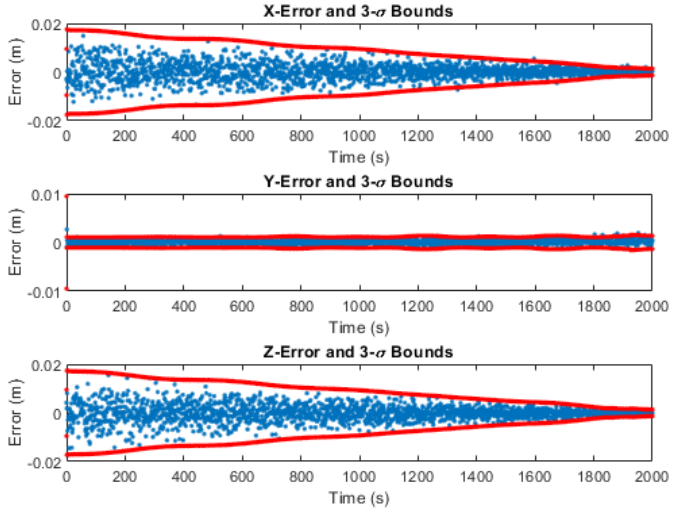

Figure 5.15: Trajectory 2 Position Error

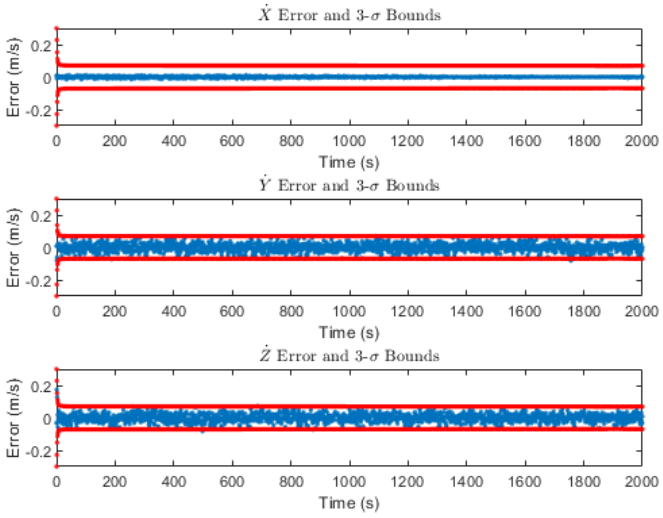

Figure 5.16: Trajectory 2 Velocity Error

The estimation error appears to be rather similar to that in 5.1, but with slight variations in the covariance due to the shifting range as the chaser moved laterally.

The orientation and angular velocity plots, seen in Figures 5.17 and 5.18 are also similar to Figures 5.5 and 5.6. Because the chaser maintains its own orientation relative to the target, markers do not drop out of frame when it moves laterally.

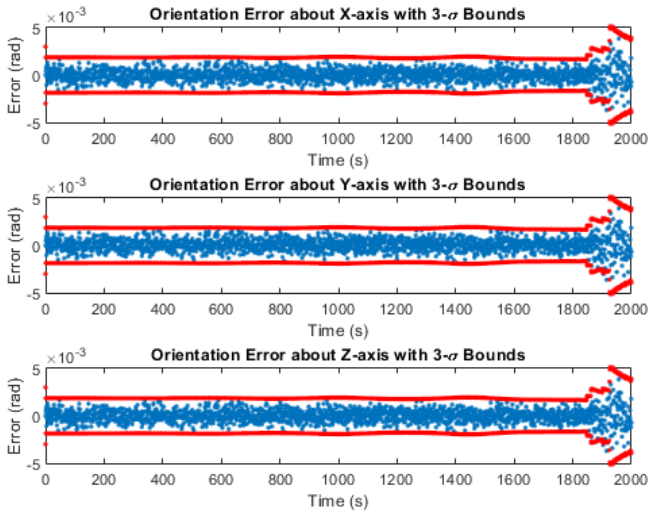

Figure 5.17: Trajectory 2 Orientation Error

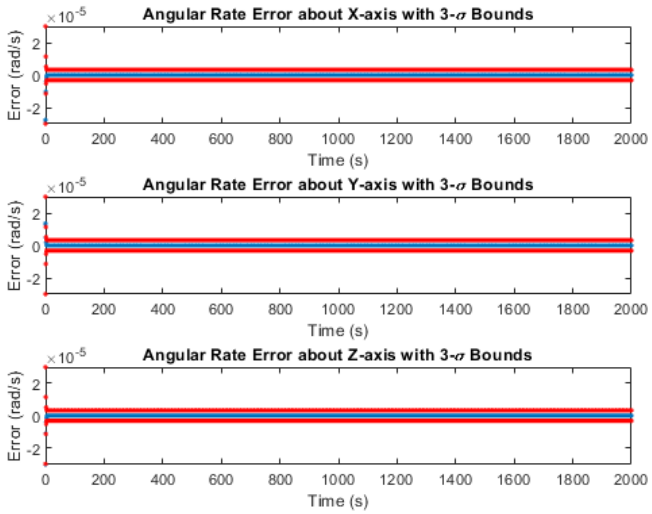

Figure 5.18: Trajectory 2 Angular Velocity Error

The bias estimate, 5.19, is actually a bit closer in this case, mainly

due to the fact that its directly computed rather than backed out from the Clohessy-Wiltshire equations.

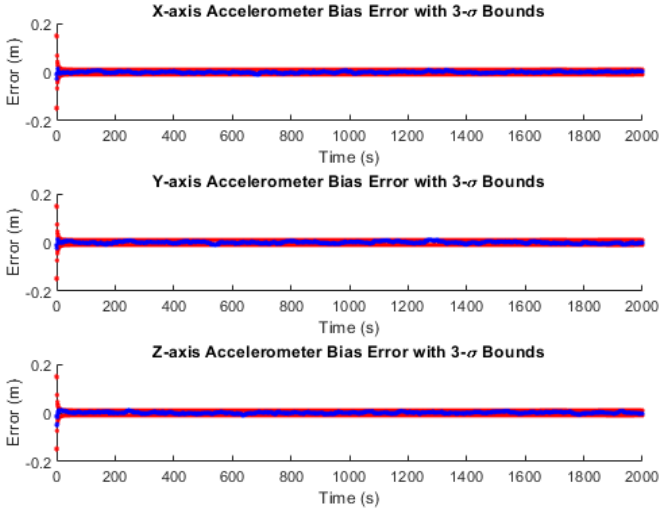

Figure 5.19: Trajectory 2 Accelerometer Bias Error

Finally, the innovation residuals, given in Figure 5.20, maintain the same trend as seen in Figure 5.9.

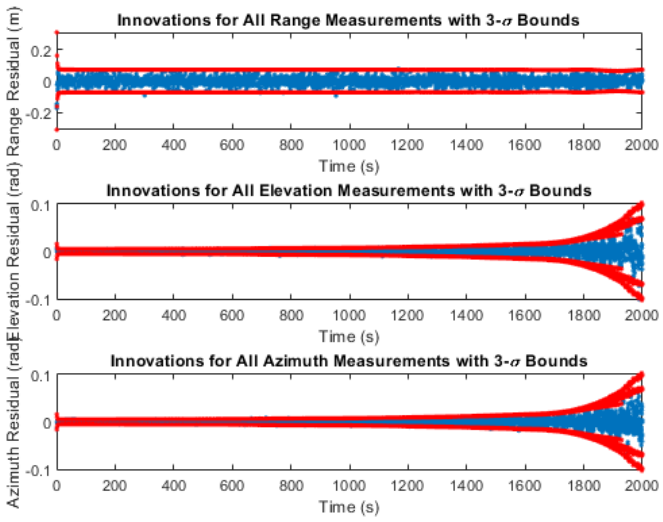

Figure 5.20: Trajectory 2 Innovation Residuals

5.4 Field of View

The measurement equation is linearize in a few places, but a major one is when the first derivative of the arctangent fuction is taken at the expected value of the state. At small angles, this approximation is nearly linear, but as the angle grows, small errors in the state will result in larger errors in the calculated derivative. As is seen in 5.21, as the range decreases, the angle between the perimeter elements, given in 2.2, grows. As far as the EKF knows, the linear approximation is perfect, so it will have an overconfidence in its own answer, leading to errors toward the end of the trajectory leaving their 3-σ bounds, as seen in 5.3 and 5.15.

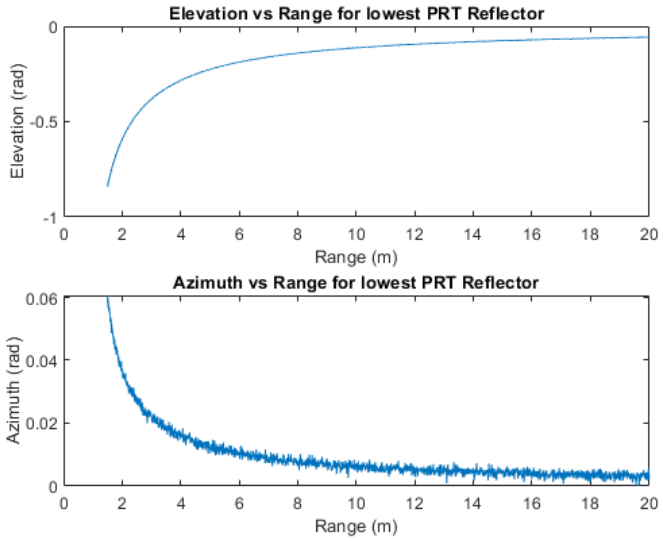

Figure 5.21: Visibility of Marker #1

To examine how this affects the number and quality of measurements on a macroscopic scale, we look to the total number of markers in view for varying fields of view, 5.22, and the effects of field of view on the residuals, 5.23 and 5.24. All simulations were run with the same RNG seed to enable an apples-to-apples comparison.

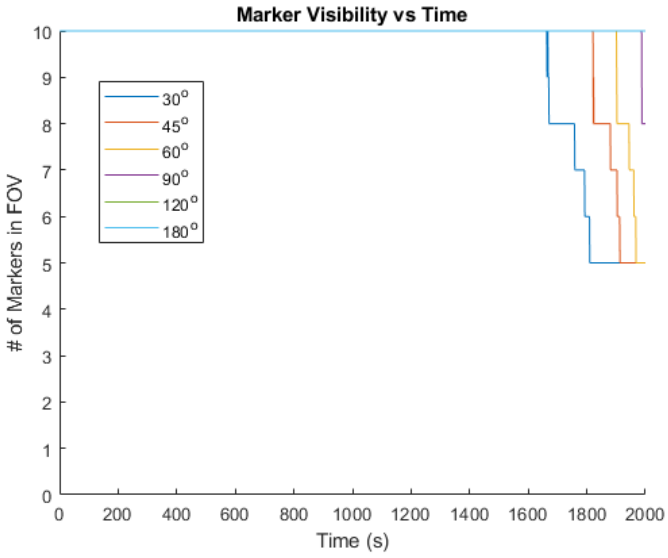

Figure 5.22: Marker Visibility for Varying Fields of View

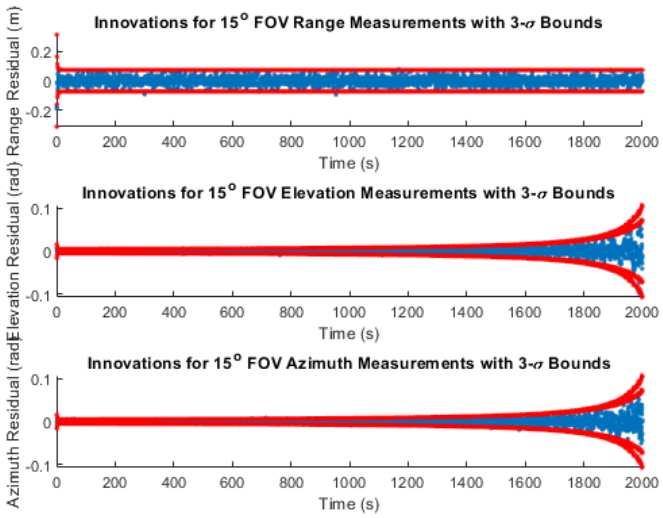

Figure 5.23: Trajectory 1 Innovation Residuals with FOV of 15^o

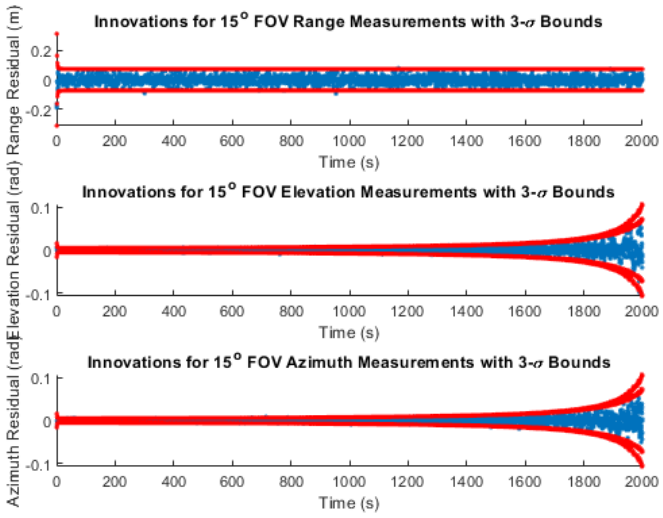

Figure 5.24: Trajectory 1 Innovation Residuals with FOV of 120^o

The fact that these have essentially the same confidence is intriguing to say the least. The expectation had been that that the estimator would be less confident in measurements with larger angles, but indeed it seems that the confidence for all three types of measurements is based mostly on range alone. However, the mean square error of both the residuals and state tell a different story.

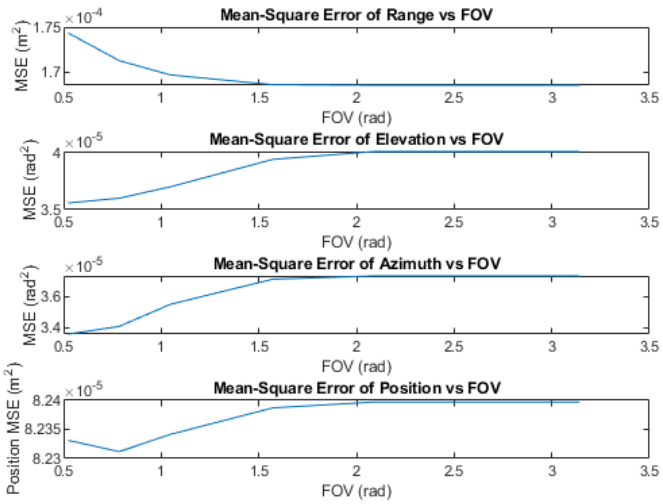

Figure 5.25: Trajectory 1 Mean Square Error of Innovations and Position Estimate

While the incorporation of the wider field of view measurements reduces the mean square range error, as expected, it actually increases the mean squared error for both azimuth and elevation. Furthermore, the mean square error of the state tends to be lower for the reduced field of view simulations, as originally predicted. Despite this, it is important to recall Figure 5.3 and the fact that the estimate improves with nearness to the target, as the information contained in even a few measurements with angles measurements around 20^o and different azimuths and elevations is far better than that contained in many measurements with at 5^o and similar azimuth and elevation.

Some other considerations for field of view, such as noise of the measurement, are not considered here, but it tends to be the case that a smaller

field of view will be better for localizing markers than a larger one for a fixed focal plane array size.

Chapter 6

Conclusion

The necessity to estimate relative position, relative velocity, target orientation, target angular velocity, and accelerometer bias was motivated in Chapters 1 and 2. In Chapter 3, the required mathematics to perform this estimation with and EKF were either derived, as in 3.4 and 3.5, or provided directly. The simulation process was discussed in detail in Chapter 4. Lastly, Chapter 5 showed the that results of the simulation lent credence to the overall approach insofar as the assumptions on dynamics and measurement model in Chapter 4 are reasonable.

A main result is that the estimator can accurately and precisely estimate the state along the a specified trajectory. The repeatibility of this estimation is validated by the Monte Carlo simulation in 5.2. Then, the estimation is shown to be sufficient for a feedback controller in 5.3. A discussion of the effects of field of view and nonlinearities occurs in 5.4.

A better estimation and control combination solution than Trajectory 2, seen in 5.3, would be an LQG controller which penalizes state errors later in the trajectory higher than those early on in the trajectory. This is future work for the lab, and such an LQG controller in a similar situation has already

been shown to work for both positiion and orientation.

The inclusion of an RGB camera and the process of moving this algorithm to a robot for hardware-in-the-loop simulation are the immediate next steps for the author and lab.

A clear direction for future work would be to simultaneously estimate the orientation and angular velocity of the chaser as well. This could either include or exclude data from a star tracker, although it is unlikely that visual odometry with respect to and unknown target a could add much information to the estimate given by a star tracker. On the other hand, extending this approach for estimating target and chaser states to the problem of SLAM with an unkown reference would be interesting and use much of the same theory from Chapter 3. In that case, both a star tracker and GPS could be used without trivializing the problem.

Bibliography

[1] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley, New York, 2001.

[2] D. M. Cho, P. Tsiotras, G. Zhang, and M. Holzinger. Robust feature detection, acquisition and tracking for relative navigation in space with a known target. In *AIAA Guidance, Navigation, and Control (GNC) Conference*. AIAA, 2013.

[3] J. Christian, H. Hinkel, C. N. D'Souza, S. Maguire, and M. Patangan. The sensor test for orion relnav risk mitigation (storrm) development test objective. In *AIAA Guidance, Navigation, and Control Conference*. AIAA, 2011.

[4] J. Christian, S. B. Robinson, C. N. D'Souza, and J. P. Ruiz. Cooperative relative navigation of spacecraft using flash light detection and ranging sensors. *Journal of Guidance, Control, and Dynamics*, 37, 2014.

[5] J. A. Christian and S. Cryan. A survey of lidar technology and its use in spacecraft relative navigation. In *LIDAR Sensors for Spacecraft Navigation I*. AIAA, 2013.

[6] J. L. Crassidis, R. Alonso, and J. L. Junkins. Optimal attitude and position determination from line-of-site measurements. *Journal of the*

Astronautical Sciences, 48, 2000.

[7] J. L. Crassidis and J. L. Junkins. *Optimal Estimation of Dynamics Systems*. CRC Press, Boca Raton, Florida, 2nd edition, 2012.

[8] H. D. Curtis. *Orbital Mechanics for Engineering Students*, chapter Relative Motion and Rendezvous. Elsevier, Waltham, MA, 3rd edition, 2013.

[9] M. Dor and P. Tsiotras. Orb-slam applied to spacecraft non-cooperative rendezvous. In *2018 Space Flight Mechanics Meeting*. AIAA SciTech Forum, 2018.

[10] X. Du, Y. He, L. Chen, and S. Gao. Pose estimation of large non-cooperative spacecraft based on extended pnp model. In *IEEE International Conference on Robotics and Biomimetics*. IEEE, Dec 2016.

[11] S. Grahn. The igla radio system for rendez-vous and docking.

[12] M. Gulino. A homotopic approach to solve the fuel optimal spacecraft proximity operations problem, May 2017.

[13] R. M. Haralick and et. all. Pose estimation from corresponding point data. *IEEE Transactions on Systems, Man, and Cybernetics*, 19, Nov/Dec 1989.

[14] International Sapce Station Partner Agencies. *International Docking System Standard Interface Definition Document*, October 2016.

[15] J. Christian J. O. Woods. Lidar-based relative navigation with respect to non-cooperative objects. *Acta Astronautica*, 126, September/October 2016.

[16] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[17] NASA. *NASA Technology Roadmaps TA 4: Robotics and Autonomous Systems*, 2016.

[18] R. Opromolla, G. Fasano, G. Rufino, and M. Grassi. A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, 93, 2017.

[19] G. Parma. Overview of the nasa docking system and the international docking system standard. In *AIAA Annual Technical Symposium*. AIAA, 2011.

[20] M. Psiaki. Autonomous orbit determination for two spacecraft from relative position measurements. *Journal of Guidance, Control, and Dynamics*, 22, 1999.

[21] J. L. Sell and et. all. Pose performance of lidar-based navigation for satellite servicing. In *AIAA/AAS Astrodynamics Specialist Conference*. AIAA Space Forum, 2014.

[22] S. Sharma, C. Beierle, and S. D'Amico. Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks. In *2018 IEEE Aerospace Conference*. IEEE, 2018.

[23] S. Sharma and S. D'Amico. Comparative assessment of techniques for initial pose estimation using monocular vision. *Acta Astronautica*, 123, June/July 2016.

[24] S. Sharma, J. Ventura, and S. D'Amico. Robust model-based monocular pose initialization for noncooperative spacecraft rendezvous. *Journal of Spacecraft and Rockets*, 55, 2018.

[25] E. Taheri, I. Kolmanovsky, and E. Atkins. Enhanced smoothing technique for indirect optimization of minimum-fuel low-thrust trajectories. *Journal of Guidance, Control, and Dynamics*, 39, Nov 2016.

[26] D. Vallado. *Fundamentals of Astrodynamics and Applications*, chapter Special Perturbation Techniques. McGraw-Hill, New York, 4th edition, 2013.

[27] J. Wang, E. Butcher, and T. A. Lovell. Ambiguous orbits in elliptic chief spacecraft relative orbit estimation with range-only measurements. *Journal of Spacecraft and Rockets*, 0, 2018.

[28] R. Zanetti. Rotations, transformations, left quaternions, right quaternions? *Journal of the Astronautical Sciences*, 01 2019.

[29] J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Transactions of the A.S.M.E.*, pages 759–768, November 1942.